Cryptanalysis of the Dedicated Hash Functions

Ali Doğanaksoy, Onur Özen, Fatih Sulak, Kerem Varıcı, Emre Yüce

Institute of Applied Mathematics,
Middle East Technical University, Ankara, Turkey,
{*aldoks, e127740, sulak, e127761, e132740*}*@metu.edu.tr*

***Abstract*—In this paper we make a survey of the recent cryptanalysis methods of dedicated hash functions which have been used to find several types of collisions for the most widely used and important hash functions, namely MD and SHA families. Starting from the well known attacks proposed by Chabaud and Joux and their successors by Biham et.al, we extend the notion by the extraordinary improvements by Wang et.al. Then, the most recent improvements by introducing automatic tools for searching collisions by Schläffer and Oswald for MD4 (and its improvement by Sasaki et.al) and de Canniere and Rechberger for SHA-1 will be introduced. Although their extensions exist for the other hash functions such as Tiger, all the mentioned attacks in this paper and their improvements are applied to MD and SHA families.**

I. Introduction

Hash functions are generally defined to take arbitrary length of input and produce a fixed length of output which is called 'fingerprint' or 'message digest' of the input message. They are used widely in cryptographic applications such as digital signatures, information authentication, redundancy, protection of passwords, confirmation of knowledge/commitment, pseudo-random string generation and key derivation [1], [2]. Main properties of a hash function that should be satisfied are listed below:

- Algorithm of a hash function should be publicly known. There may not be any secret parameters.
- A hash function gets an arbitrary length of input and produces a fixed length of output.
- For a given value x and a hash function h, it should be 'easy' to compute $h(x)$.

A cryptographic hash function, on the other hand, has to satisfy some security notion as:

Preimage Resistance: For a given value $h(x)$, it should be 'hard' to compute x.

Second Preimage Resistance: Given x and $h(x)$, it should be 'hard' to find x' such that $x \neq x'$ and $h(x') = h(x)$.

Collision Resistance: For any x it is 'hard' to find x' where $x \neq x'$ and $h(x) = h(x')$.

A common way to construct a hash function is the use of iterations [3]. Firstly a padding rule is applied to the message x. A compression function, say $h(x)$, is used at every step of iteration that gets fixed length of input and produce n-bit output. Output of every step h_{i-1} (which is called as chaining variables or intermediate variables [2]) is used in the next step together with the message x_i and produces h_i. Last step of iteration, that is h_t, is the hash value of the message. The first chaining variable, which is called IV (Initial Value), is often taken fixed. Equation below describes the hashing procedure.

$$h_0 = IV, h_i = h(x_i, h_{i-1})\ i = 1, 2, ..., t, H(x) = h_t$$

There are various construction methods. The most widely used method is the *Merkle-Damgärd Paradigm* [4], [5] which proves that if the design is built on Merkle-Damgärd Paradigm then the security of hash function relies on the security of the compression function [4]. So, the first step of attacking a hash function is to find weaknesses in compression function rather than dealing with the whole hash function, if the design assumes Merkle-Damgärd Construction Principle. The most popular designs built on Merkle-Damgärd Principle are MD4, MD5, SHA-0 and SHA-1.

Recent years have witnessed very important improvements on the cryptanalysis methods of dedicated hash functions which have been used to find several types of collisions for the most widely used and important hash functions, namely MD and SHA families. One of the common features of the attack methods is that they are all in differential nature. The well known attack proposed by Chabaud and Joux [6] is considered as the commencement of the application of differential cryptanalysis to find collisions. Then, Biham et.al in [7], [8] improved the attack of Chabaud and Joux by introducing new concepts such as neutral bits.

The attacks proposed by Wang et.al.[9], [10], [11], [12] brought the attentions of the cryptography researchers immediately. What is important about the attacks of Wang et.al is that they use a new method called *message modification* to decrease the compleexity of the collision attacks. However, some righteous cirtics have been made about the methods proposed by Wang et.al and a new concept was born to search differential paths by the use of automatic tools. The most recent improvements by introducing automatic tools for searching collisions by Schläffer and Oswald for MD4 (and its improvement by Sasaki et.al) and de Canniere and Rechberger for SHA-1 are considered as the most valuable tools nowadays.

The remainder of the paper is structured as follows. The short descriptions of the hash functions MD4, MD5, SHA-0 and SHA-1 are given in section 2. Section 3 covers the methods of Chabaud, Joux and Biham et.al. We will introduce the attack methods proposed by Wang et.al for finding collisions in section 4 and the 5^{th} section is devoted to the automatic tools for finding collisions. Although their extensions exist for the other hash functions such as Tiger, all the mentioned attacks in this paper and their improvements are applied to MD and

SHA families. We conclude the survey in the last section.

II. The Brief Descriptions of MD4, MD5, SHA-0 and SHA-1

A. MD4 and MD5

Many of the dedicated hash functions are based on the design principles of MD4 which will be described throughout this subsection where the differences between MD4 and MD5 are also detailed. MD4[13] was designed by Ron Rivest in 1990 which takes at most 2^{64} bits of input, as its padding rule permits, and produces 128-bit fingerprint of it. It is also built on the Merkle-Damgärd Principle. The successor of MD4, MD5[14] was published again by Ron Rivest two years later.

MD4 takes message blocks of 512 bits and produces 128-bit output. In order to make the message an exact multiple of 512-bit, the padding procedure is applied. First, one 1 bit and enough 0 bits are added to the end of the message to make the length 448 modulo 512. Finally, 64-bit representation of the original message length is filled for the remaining 64 bits. IV value of the MD4 is taken as: A_0: 0x01234567, B_0: 0x89$ABCDEF$, C_0: 0x$FEDCBA$98, D_0: 0x76543210

MD4 has 3 rounds, each consisting 16 steps. In every step i, the state variables A_{i-1} , B_{i-1} , C_{i-1} , D_{i-1} ,the message word M_i , the step constant K_i and the shift value s_i are used. A_{i-1} , B_{i-1} , C_{i-1} , D_{i-1} are intermediate variables which are updated at every step. M_i is one of the 32-bit part of the 512-bit message which is determined by a permutation at each step. K_i and s_i are appropriate constant and shift values respectively. The figure below visualizes one step of the compression function of MD4.

Every round of MD4 compression function uses a different non-linear boolean function as stated below:

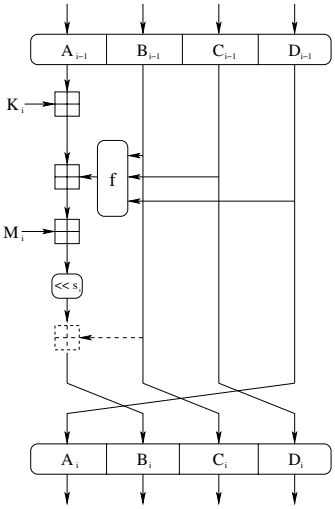
$$\begin{aligned} F(X,Y,Z) &= X \wedge Y \vee \neg X \wedge Z \\ G(X,Y,Z) &= X \wedge Y \vee X \wedge Z \vee Y \wedge Z \\ H(X,Y,Z) &= X \oplus Y \oplus Z \end{aligned}$$

Above $\wedge$, $\vee$ and $\oplus$ mean bitwise AND , OR , XOR respectively. $\neg X$ shows the negation of X and"$<< s$" is used for shifting the bits to the left by s bits. Exact values of constants and shift values are given in [13].

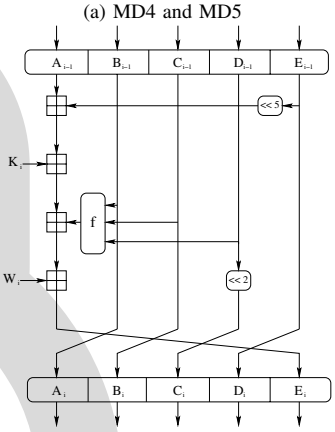
In 1992 Ron Rivest made some refinements on MD4 algorithm and published the new version as MD5. These refinements include the addition of the fourth round as $I(X,Y,Z) = Y \oplus (X \oplus \neg Z)$ and the change of the second round (that is $G(X,Y,Z) = X \wedge Y \vee X \wedge Z \vee Y \wedge Z$) to $G(X,Y,Z) = X \wedge Z \vee Y \neg Z$. Besides, the state variable B_i is added to the output of the boolean function at each step. The full list of changes including the constants and the shift values are given in [14].

B. SHA-0 and SHA-1

SHA−0 (Secure Hash Algorithm) [15] is published as a Federal Information Processing Standard (FIPS) by U.S. National Security Agency in 1993. After two years, expansion method of SHA-0 was changed and republished as SHA-1 [16] by just modifying the message expansion. SHA is mainly inspired



(a) MD4 and MD5



(b) SHA0 - SHA1

Fig. 1: MD-SHA Family Hash Functions

from MD Family. MD4, SHA−0 and SHA−1 takes at most 2^{64} bits of message block; however the output of SHA−0 and SHA−1 is 160 bits. Output is the concatenation of 5 words of 32 bits. It has 5 rounds of 16 steps each.

SHA−0 is mainly constructed over MD4 structure. Firstly the padding rule, same as MD4, is applied to the message to make the message an exact multiple of 512-bits. A 512-bit part is taken from the message as input. Message expansion, which is one of the main differences from MD4, is applied to the 512-bit part. Using the notation in [15], message expansion for SHA−0 is defined as follows:

$$W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}, \ \forall i, 16 \leq i < 80$$

After the message expansion, 512-bit input (that is $< W_0, W_1, ..., W_{15} >$) becomes $< W_0, W_1, ..., W_{79} >$. As another difference from MD4, there are five 32-bit words namely $< A_i, B_i, C_i, D_i, E_i >$ as intermediate variables in SHA−0. IV of SHA−0 is taken as:A_0 = 0x67452301, B_0 = 0x$EFCDAB$89, C_0 = 0x98$BADCFE$, D_0 = 0x10325476, E_0 = 0x$C3D2E1F$0. The figure above better describes the step operations used in SHA-0.

SHA−1 is very similar to SHA−0. The only difference between SHA−0 and SHA−1 is the message expansion. Message expansion in SHA−1 just makes use of a simple rotation [16]:

$W_i = ROL_1(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16})$, $\forall$ i, $16 \leq i < 80$.

III. Cryptanalysis of MD4, MD5, SHA-0 and SHA-1

The objective of this paper is to overview the latest attack scenarios for finding collisions to the dedicated hash functions. In order to find a collision randomly for an n-bit hash function without knowing the internal structure, the attack works with probability 2^{-n}. However, by using the birthday paradox one needs just $2^{-n/2}$ pairs to get a collision. Instead of finding collisions randomly, one can use the internal structure of the hash function attacked. The latest attacks are similar in that they are all differential attacks in nature and each has its own characteristics and properties to construct differential paths special to it. In order differential attacks to succeed the number of message pairs needed should be kept under the birthday paradox limit.

A. Differential Collisions in SHA and The Concept Neutral Bits

B. The Method Proposed by Wang et.al

1) The Overview of the Attack: This chapter fully covers the approach of the attacks by Wang et.al that have been applied to many dedicated hash functions recently. It was first applied to `MD4` [9], which is a basis for many used hash functions besides `MD`-family, and then extended to the `MD` and `SHA`-family hash functions with some modifications[9], [12], [10], [11], [8]. Their attack is a differential attack modified for the hash functions that is trying to find two colliding pair of messages.

Wang et.al used the additive difference $\Delta M = M - M'$ instead of XOR difference between meassages and made an extensive use of the so called *local collision* (or *internal collision*) which is a range of steps with a zero difference at the beginning of the first and the end of the last step, but with nonzero differences in the intermediate steps [17]. These non-zero differences are edited by message words to get and cancel the differences during the local collision steps. Changed message words during the local collision affects the other steps of the hash function generally and have to be cancelled for these steps in order to get a full collision. This cancellation process affects the complexity of the attack dramatically and is done by using the properties of the boolean functions applied on the corresponding steps of the compression function. For each step, some conditions on the corresponding bits have to be imposed in order to cancel the differences edited by the local collision. Finally, in the message modification steps the complexity due to the conditions on bit values is reduced by some sort of smart tricks. The method proposed by Wang et.al can be summarized as follows:

1) Find a message difference using the local the local collision working with a high probability.
2) Obtain the differential path for the given message difference. This step contains the conditions on the bit values in each step.
3) By message modification techniques, modify the second message M' by which the differential path fulfills the necessary conditions to reduce the complexity.

The rest of the section explains the application of the above attack by Wang et.al to the hash functions `MD4`, `MD5` and `SHA-0`.

2) Cryptanalysis of MD4: The collision search for `MD4` is the direct application of the above attack and does not contain any further details special to `MD4`. Since it is a basis for many dedicated hash functions, the cryptanalysis of MD4 is important. In the first step of the collision search attack, the message difference $\Delta M = M - M'$ between two messages M and M' is determined using the local collisions. These local collisions can be included in one of the three rounds in `MD4` but so as to make it work effectively(that is probabilistically), the round using XOR as boolean function is used. Besides, it is shown in [9], [17] that it is the best choice for the full collision search for `MD4`.

Searching for a local collision is a straightforward manner for `MD4`. However, some tricky details should be kept in mind. Wang et.al preferred to impose differences on message words during local collisions by perturbing less message words in order not to deal with many message words in the other steps. They found a 6-step local collision with probability 2^{-2} by changing just 3 message words. However, Sasaki et.al [18] improved this probability to 2^{-1} by finding a 5-step local collision that changes 5 message words which is proved to be the best probability for `MD4`. The former used the message difference : $\Delta m_1 = 2^{31}$, $\Delta m_2 = 2^{31}$ 2^{28} , $\Delta m_{12} = 2^{16}$. This clearly implies that there exists a small local collision between steps $33-38$ and a large collision between steps $0-24$ to get a full collision. The reason why the rounds $33-38$ is chosen (the message words Δm_1, Δm_2, Δm_{12}) is given in [9], [17] (for the attack mounted by Sasaki et.al in [18].

After the message difference has been obtained by the local collision introduced in the third round, a differential path for this message difference needs to be found. In the first part, the message words m_1, m_2 and m_{12} have been changed. In order to get a full collision, the differences introduced by the message words in these steps have to be cancelled out. Namely, a new local collision between steps $0-24$ needs to be found.

The cancellation step through 25 steps is not an easy task. Since all possibilities of the differences are very high, the uncontrolled propagation of differences should not be allowed. Moreover, since the differences of message words are determined by the local collision in previous step, it is not allowed to change the differences caused by the local collision. Instead, the controlled propagation of the differences through boolean functions in the first two rounds can be used by adding conditions on bit values and expanding the message differences. This will lead to a new local collision for the first 25 steps of the compression function of `MD4`. A complete differential path can be found in [9], [17].

The second part of the attack imposes conditions on bit values in every step of the compression function and clearly since there exist 125 conditions on bit values [9], for a random message, finding a collision requires 2^{125} message pairs which is very high compared to birthday attack. However, Wang

et al. applied a new method called *message modification* to reduce this complexity to 2^6. This complexity was then reduced to 2^2 by Sasaki et.al [18]. By taking an arbitrary message and modifying it corresponding to the conditions imposed on bit values and the differential path, a new modified meassage is obtained without considering too many message pairs. Message modification is composed of three major parts [17], [9], [12] :*Single Step Message Modification*, *Multi Step Message Modification* and *Advanced Message Modification*.

In single step message modification, the conditions imposed on the first round are fulfilled by inverting the step operations or just by flipping the message bits (if multi step message modification is needed). For every condition on the register which is not fulfilled, the corresponding bits of that register are corrected by flipping those bits and then the new message word is computed by using this new modified register value by reversing the step operation. The cost of this operation is negligible.

If multi step message modification is needed, then single step modification and the multi step modification can be done interleavingly by flipping the bits of the messages obeying both the conditions on register values imposed in round one and two but not disturbing the differential path. That is, some bits of the message word M_0 can be flipped interleavingly in order to satisfy the conditions on both A_0 and A_{16} [17]. However, some other conditions might have been imposed by reference not just by value (i.e. $A_{0,1} = A_{1,1}$). Then, the corresponding message word has to be changed and the register values have to be computed again by reversing the step operation in the first round. But this time, the register value A_0 has to be corrected again. Therefore, in rounds$(1-4)$ where A_0 is used, corresponding message words have to be changed again. Now, a multi step message modification is needed for the message words in the corresponding rounds.

Advanced message modification is used because it is not possible to correct all conditions of step 18 using the same technique, because there are already conditions at the corresponding bit positions of M_8. Thus, a new method called advanced modification technique is used. The details of this method are not covered in the content of this paper, for the details please refer to [17].

3) Cryptanalysis of MD5: This subsection is dedicated to the cryptanalysis of `MD5` which is the most used hash function in various applications. The attack proposed by Wang et.al [12] can find many real collisions which are composed of two 1024-bit messages (M_0, M_1) and (M'_0, M'_1)). Instead of using one block message with one iteration, to break `MD5`, Wang et.al made extensive use of the so called *near collisions*. At the end of the first iteration, an intermediate hash value with a difference in few bits is found with a high probability and by using the structure of Merkle-Dämgard paradigm, after the second iteration the second difference is cancelled and the full collision is found. The first iteration succeeds with probability 2^{-37} and the second with probability 2^{-30}.

Following the notation used in [12] message differences are selected as: $\Delta H_0 \overset{M_0,M'_0}{\rightarrow} \Delta H_1 \overset{M_1,M'_1}{\rightarrow} \Delta H = 0$ where

$$\Delta M_0 = M'_0 - M_0 = (0,0,0,0,2^{31},0,0,0,0,0,0,2^{15},0,0,2^{31},0)$$
$$\Delta M_1 = M'_1 - M_1 = (0,0,0,0,2^{31},0,0,0,0,0,0,2^{15},0,0,2^{31},0)$$
$$\Delta H_1 = H'_1 - H_1 = (2^{31}, 2^{31}+2^{25}, 2^{31}+2^{25}, 2^{31}+2^{25}).$$

The attack procedure can be explained as follows as in the case of `MD4` [12], [19]:

1) (a) Take a random message M_0.
   (b) Modify M_0 by message modification techniques to reduce the overall complexity and find M'_0 with probability 2^{-37}.
   (c) Test whether all the differential path holds.
2) (a) Take a random message M_1.
   (b) Modify M_1 by message modification techniques to reduce the overall complexity and find M'_1 with probability 2^{-30}.
   (c) Check whether the message pair collides.

It is claimed that completing the first iteration successfully takes about 2^{39} `MD5` operations and the second step takes about 2^{32} `MD5` operations.

4) Cryptanalysis of SHA-0: After the improvements on the collision search attacks on `MD4` and `MD5` proposed by Wang et.al, the application of this attack to the other dedicated hash functions seemed to be straightforward. On the other hand, the *message modification steps* which are the most crucial parts of the collision search attacks in this section do not seem to be applied immediately due to the structure of the message expansion algorithms used in `SHA-0` and `SHA-1`.

In the first attempt, Wang et.al again used local collisions together with some constraints on the message differences in order to find full collisions. For each round of the compression function of `SHA-0`, there exist a 6 step local collisions with different probabilities depending on the boolean function. What is important about the local collisions is the starting step which is generally shown by the *disturbance vector*. It is easy to show that the disturbance vector satisfies the same recurrence relation used in the message expansion algorithm which is linear and of degree 16. Therefore, the first 16 message words (similarly disturbance) determine the others. Wang et.al imposed some conditions on the message differences as follows to get an efficient differential path[10].

1) The difference of the last 5 expanded message have no difference in order to get a collision ($\Delta W_i = 0$ for $i = 75, 76, 77, 78, 79$).
2) $\Delta W_i = 0$ for $i = -5, ..., -1$ to avoid truncated collisions in first few steps.
3) Avoid to have consecutive ones in order to get rid of an impossible collision due to a property of IF.

When the above conditions are imposed, only 3 of the 2^{16} possible are left. Given the disturbance vector x, $hw_{r+}(x)$ is defined to be the hamming weight of x from step r to 80. To minimize the complexity of the collision search, one of the vital conditions is to keep $hw_{17+}(x)$ as small as possible because of the message expansion algorithm. The corresponding vector has $hw_{17+}(x) = 27$ and the complexity of the collision search attack is 2^{58}. Existing techniques to

improve attack are limited as they can not provide anything about the message modification steps. Therefore, the strategy should be changed for searching collisions in SHA-0.

New techniques for searching collisions in SHA-0 contain some relaxation on the conditions of the message differences given above together with a variation of the message modification for the steps $17-20$. Namely, to get full collision the first condition should remain, the other two conditions can be removed such that there exist many good disturbance vectors with small $hw_{17+}(x)$. Among the 2^{16} vectors satisfying the first condition, 30 of them have $17 \leq hw_{17+}(x) \leq 19$ and 3 of them have hamming weight 3 in round 3. This observation is important due to the fact that there exist small message differences and this makes the message modification easier [10].

By this new type of disturbence vectors, Wang et.al succeeded to find collisions for SHA. They present the first attack on the full SHA−1 with complexity less than 2^{69} hash operations. This attack is also available to find one-block collisions for the SHA−1 reduced variants less than 76 rounds [11].

IV. Automatic Tools Searching for Differential Paths

The attacks introduced in the previous chapter brought an extraordinary interest from the cryptography researchers. However, in spite of these developments, there have been many critics about the attacks due to the claims by Wang et.al that all those differential paths and the necessary conditions had been found by intuition and hand. Therefore, some researchers spent some time on proving the attacks introduced by Wang et.al. In fact, there was no doubt about the truth of the collision search, but the suspects about these attacks gave birth to the automatic tools searching for the differential paths. The first automatic tool has been generated by Schläffer and Oswald [20] in order to find collision for MD4. Given the message differences the algorithm finds the differential path together with the necessary conditions. This tool was then improved by Sasaki et.al [18] by accepting any message difference to find the differential path. In Asiacrypt '06, Rechberger and de Cannière [21] developed an algorithm to find differential paths for SHA−1 which is now used for finding differential paths for SHA family. This section is devoted to these 3 automatic tools.

A. Schläffer et al.'s Algorithm

In the Wang et al.'s collision attack to MD4, after introducing a local collision and the message difference, a suitable differential path was given directly without any explanation. Therefore, some algorithms were developed in order to obtain differential paths whenever a local collision was given. One of the most important algorithm was developed by Schläffer and Oswald [20]. They used Wang et al.'s message difference and found a differential path that cancels the introduced message difference in the first two rounds of MD4.

The defined algorithm finds a collision in three main steps.

- Target Difference Computation Step
- Cancellation Search Step
- Correction Step

In the first step, the propagation of the introduced message difference is computed forward and backward through MD4 algorithm. The aim of this step is to derive the necessary differences in each step and cancel out the message difference. The forward computation of the message difference in each step is called *disturbance difference* and the backward computation is called *correction difference*. *Target differences* are obtained by adding these two differences to get the necessary difference for the output of the boolean function.

Target differences are tried to be cancelled by f_i in the cancellation search step. However, in each step i, all elements of the target differences may not be cancelled. Therefore, all variations of the target differences have to be considered and checked. The cancelled elements of the target differences are removed and the remaining ones are restored to be cancelled in the later steps.

The carry expansions of the input differences of f function are used generally to cancel the elements of the target differences. In fact, carry expansion is limited to keep the complexity under control and at the end of cancellation search step, the conditions are determined with the aid of the used carry expansions and mostly some contradictions can occur in the whole path. In order to solve this contradiction, next(Correction) step begins in the algorithm.

In the correction step, contradictions are fixed in the obtained path. If it is not possible, some additional differences are introduced in the previous steps to cancel the contradicted elements and as a result of this algorithm, lots of new differential paths found. The diagram of the Schläffer et al.'s algorithm of better describes the procedure.

B. Sasaki et al.'s Algorithm

A new message difference for MD4 is introduced in FSE 2007 by Sasaki et.al[18]. It is also based on finding two local collisions in MD4 one is in the third round and the other in the first and second round. But this time, just one sufficient condition is used when producing a local collision in the third round. After deciding the beginning and the end points of the local collisions, searching for a differential path part commences. In order to do this, the above algorithm had been tried first, but constraints in the algorithm for less complexity caused also a reduced search space and did not give any result. Thus, a new differential path search algorithm was defined. This algorithm provides a larger search space with using f function more effectively in the cancellation part. It has three main parts.

- **Forward Search:** From the first step to the fourth step *forward search* is performed and propagation of the f function is controlled in a search space as large as possible. After the fourth step, possible differences are called as *Potential Differences*.
- **Backward Search:** From the sixteenth step to the eighth step, *backward search* is performed. Target differences are used in this step but some changes are made in its definition and performed from step 8 to 16. The difference in the chaining variables are fixed after the 16^{th} step

(x_i, x_i')	(0,0)	(1,0)	(0,1)	(1,1)
?	❖	❖	❖	❖
–	❖			❖
x		❖	❖	
0	❖			
u		❖		
n			❖	
1				❖
#				

(x_i, x_i')	(0,0)	(1,0)	(0,1)	(1,1)
3			❖	❖
5		❖		❖
7		❖	❖	❖
A	❖		❖	
B	❖		❖	❖
C	❖	❖		
D	❖	❖		❖
E	❖	❖	❖	

Fig. 2: Generalized Characteristics

and target differences are calculated from the 16^{th} step to the 12^{th} step. After obtaining the chaining variable differences, the same operation is performed again from the steps 12 to 8 step. The obtained differences are called *aimed differences*.

- **Joint Algorithm:** In this step, potential differences are calculated from the fourth round to the eighth round and checked whether there exist a match in aimed difference part or not. The potential differences that have a match in aimed difference are kept and tried to be cancelled in the following steps by using carry expansions in the input differences of f function. If all differences can be cancelled, then joint algorithm is finished and a new differential path is found.

C. Rechberger and de Cannière's Algorithm

In classical differential cryptanalysis, just differences and the differential path is considered. First, in their attacks Wang et.al extended the difference and the differential notion to the case of *signed bit* representations or differences. In their notation the bit value is set to 1 if the change $0 \rightarrow 1$ occurs and -1 if $1 \rightarrow 0$ occurs [9] which leads to a better attack scenario. Both are same in the XOR differences but they are very important in modular differences.

Rechberger and de Cannière furher extended this representation by introducing the so called *the generalized characteristics*. The overall table is shown below [21]. In generalized characteristics, every bit of the chaining variables and the extended message words is assigned to a generalized bit representation. In this representation, one can impose conditions although there exist a zero difference.

What is important about the developed algorithm is the cost of finding the suitable message pairs that collide. It is done by estimating the number of nodes in a search tree. In [21], it is shown that imposing conditions on the state variables after round 16 affects the work factor of algorithm dramatically. The generalized characteristics and the observation made above are the necessary tools for constructing the differential path.

In order to construct the differential path, determining when and where to add which condition, letting conditions propagate and avoiding inconsistent conditions [21] are the tricky points. Consistency and the propagation of conditions are done by adding the conditions on the bits of the expanded message words and checking the consistency with the state variables through the differential path. Determining the conditions, on the other hand, are done by a greedy approach by trying all the conditions to get the most efficient one. However, some of the bits should be kept unrestricted. Those differences are taken as zero-difference to reduce the search complexity.

The best method to get the message difference is to obtain a local collision. However, it is not known where to get a local collision. Rechberger and de Cannière minimized the differences occured between steps 40 and 59 because the propagation of differences through the majority function are inconsistent. Therefore, the rounds with the XOR function which have little perturbations in round 3 are used to get a local collision. After constructing the local collision, the methods described above are used to construct the differential path.

V. Differential Collisions in SHA-0

Chabaud and Joux developed a new attack on SHA-0 in 1998 ([22]). The attack is similar to differential cryptanalysis. To find a collision for SHA-0 one has to deal with expansion function E and nonlinear parts of the algorithm, that is f functions and ADD operation.

A. Dealing with expansion function

The attack will be performed step by step. First consider a hash function SHI1 which is obtained from SHA-0 by replacing ADD and f by XOR functions. (W^i denote the ith word of the expansion for $0 \leq i < 80$ and the bits of one word is numbered as W_k^i for $0 \leq k < 32$.)

First step of the attack is complementing one bit of W^i. If W_1^i is complemented this bit will affect the A_1^{i+1}. As it can be seen from the SHA-0 diagram, this change will modify B_1^{i+2}, C_{31}^{i+3}, D_{31}^{i+4} and E_{31}^{i+5} considering rotations. Thus negating W_1^i, W_6^{i+1}, W_1^{i+2}, W_{31}^{i+3}, W_{31}^{i+4} and W_{31}^{i+5} will result two different paths from $\{A^i, B^i, C^i, D^i, E^i\}$ to $\{A^{i+6}, B^{i+6}, C^{i+6}, D^{i+6}, E^{i+6}\}$ and gives a local collision.

Since every operation is linear local collisions can be applied simultaneously and two different paths from $\{A^0, B^0, C^0, D^0, E^0\}$ to $\{A^{80}, B^{80}, C^{80}, D^{80}, E^{80}\}$ can be obtained easily. In these paths first one uses original W and the second one uses a modified one which will be denoted by W'. The local collisions should be chosen such that both W and W' are the outputs of the expansion procedure.

An error vector m_0 can be constructed for local collisions. m_0 consists of 80 bits and the ith bit of m_0 is 1 if W_1^i is negated, 0 otherwise. Obviously W_1^i cannot be negated for $i \geq 75$ as a perturbation is corrected after 6 rounds. $M_0 = \langle M_0^{-5}, M_0^{-4}, \ldots M_0^{79} \rangle$, which is a perturbative mask on W is defined by:

$$\begin{aligned} M_0^i &= 0; \qquad for\ -5 \leq i \leq -1 \\ M_{0,k}^i &= 0; \qquad if\ k \neq 1\ for\ 0 \leq i \leq 79 \\ M_{0,1}^i &= m_0^i; \qquad for\ 0 \leq i \leq 79. \end{aligned}$$

The first correction mask is obtained by $M_1^i = ROL_5(M_0^{i-1})$ using SHA-0 operations. Similarly $M_2^i = M_0^{i-2}$, $M_3^i = ROL_{30}(M_0^{i-3})$, $M_4^i = ROL_{30}(M_0^{i-4})$ and $M_5^i = ROL_{30}(M_0^{i-5})$. So the global differential mask $M^i = M_0^i \oplus M_1^i \oplus M_2^i \oplus M_3^i \oplus M_4^i \oplus M_0^i$ should be an output of E_0.

All masks M_k should satisfy the equation (1). This condition is fulfilled if the initial perturbative mask satisfies

$$M_0^i = M_0^{i-3} \oplus M_0^{i-8} \oplus M_0^{i-14} \oplus M_0^{i-16},\ 11 \leq i < 80.$$

Then it remains to find masks that satiffy this equation.

B. Dealing with f functions

The second step of the attack is the influence of nonlinear f functions. Consider a hash function SHI2 which is obtained from SHA-0 by replacing ADD by XOR function. f function changes with round and can be XOR, IF and MAJ. In XOR case there will not be any problem but IF and MAJ functions should be detailly analyzed. Assume that a transition occurs in $f(B^i, C^i, D^i)$. There are four cases:

1) There is no change in the inputs.
2) There is a single difference in the entries of bit 1 of B^i.
3) There is a single difference in the entries of bit 31 of C^i or D^i.
4) There are two differences in the entries of bit 31 of C^i or D^i.

For MAJ function the probabily that output will not change is 1 for case 1 and 1/2 for the other cases. For IF function this probability is 1 for case 1, 1/2 for cases 2 and 3 and 0 for case 4. The details of calculations can be found in [22]. So a perturbation pattern should be chosen such that there will not be any adjacent perturbations for the IF rounds. At this point a collision search should be made using computer. The probabilities will give the complexity of the attack. A sample perturbation which satisfy the given conditions is given in [22].

C. Dealing with ADD function

The third step in the attack is to deal with addition function. Consider a hash function SHI3 which is obtained from SHA-0 by replacing f by XOR function. The problem in this structure is the carry bits in perturbations. Each correction and each perturbation may lead to a carry bit but if a perturbation is applied to W_1^i (and should be applied in order to get a good complexity) three corrections on bits 31 (W_{31}^{i+3}, W_{31}^{i+4} and W_{31}^{i+5}) will be obtained. But there cannot be any carry from 31st bit. In this case the probability for each perturbation is $1/2^3$.

It is possible to improve this probability. Assume that W_1^i changes from 0 to 1 (an incrementation). By simple computation it can be seen that W_6^{i+1} should be 1 to get rid of carry bit. An incrementation should be corrected by a decrementation and vice versa. The probability is 1/2 for no carry and 1/2 to ensure that XOR keeps the change in the same direction. By using all these facts, a pattern is found with the probability of $1/2^{44}$ in [22].

D. Attack for SHA-0

The attack on SHA-0 is the composition of three attacks on SHI1, SHI2 and SHI3. All perturbations should be chosen such that there will not be any carry bits and IF and MAJ functions should be deeply analyzed. For IF function the attack is similar to SHI2. Case 4 should be avoided. For case 2 the probability is again 1/2 but for case 2 direction should be preserved and therefore the probability is 1/4.

For MAJ function the direction cannot be reversed so the probabilities for case 2 and case 3 is still 1/2. There are some new conditions for case 4 and under the new conditions probability is 1/2. Using all constraints a pattern with probability $1/2^{61}$ is given in [22]. Therefore the complexity of the attack is 2^{61} which is better than birthday attack. This attack cannot be applied to SHA-1 under the same conditions because of the new design structure of expansion function in SHA-1.

E. Near Collisions of SHA-0

In 2004 Biham and Chen improve the Chabaud-Joux attack ([23]). They propose an efficient search algorithm and they find near-collisions of SHA-0. For a hash function near-collision resistance is also a security criteria. For any x it is 'hard' to find x' where $x \neq x'$ and $h(x)$ differs from $h(x')$ in only a small number of bits.

In this attack a disturbance vector $\mathtt{D}$ is used. It is defined similarly; if there is a disturbance in a round, the corrosponding entry of the vector is 1 and 0 otherwise. This vector should have low Hamming weight in order to get low attack complexity. If all the corrections succeed $A_{i+1} \oplus A'_{i+1}$ should be equal to $\delta = \mathtt{D}_i << 1$ where $<<$ denotes the shift.

The main idea in the attack is to start collision search from an intermediate round. Two definitions are needed to understand the attack.

Definition V.1. *If $A_i \oplus A'_i = \delta_i$, $\forall i \in \{1, 2, \ldots, r\}$ then it is said that a pair of message conforms to δ_r.*

Definition V.2. *Let M and M' be a pair of messages that conforms to δ_r for some $r \geq 16$. A set of bits $S \subseteq \{0, 1, \ldots, 511\}$ is called neutral with respect to M and M' if for all pairs of messages received by complementing any subset of bits in S in both messages M and M' also conforms to δ_r.*

The size of maximal 2-neutral set is denoted by $k(r)$. Two additional properties given for the collision search:

1) The message pairs conforms to δ_r.
2) The message pair has large 2-neutral set of bits.

To find a collision last five entries of $\mathtt{D}$ should be 0. But this conditions need not to be satisfied when near-collisions are searched. r should be selected such that $2^{k(r)} \geq 1/p(r)$ should be satisfied, where $p(r)$ is the probability of successful corrections in all the rounds that conform to δ_r. Actually, r should be selected as the largest number that satisfies the given equation.

An algorithm is given in [23] to find 2-neutral set of bits. By the help of the algorithm a collision is found for 65 rounds of SHA. For the 82 rounds extended SHA a collision is found with 2^{-71} probability. By taking $r = 22$ near collisions are found with 2^{-43} probability. This attack also cannot be applied to SHA-1 directly due to expansion function.

VI. Conclusion

In this paper, we investigated the recent developments in finding collisions for the well known cryptographic hash

functions, namely the MD and SHA family hash functions. The table below briefly compares those attacks with their complexities. Although it is not covered in the content of this paper, discussed methods can be easily adapted to the other hash functions. Before the NIST competition concerning the design of a new hash function as a standard, we believe that this survey will be a very helpful tool to the cryptographic community.

Hash Function	Hash Size	Collision	Attacks	Complexity
MD4	128	Yes	2004, Wang et al.[9], 2007, Sasaki et al. [18]	2^6 2^2
MD5	128	Yes	2005, Wang et al. [12] 2006, Black et al. [19]	2^{39} 2^{30}
SHA-0	160	Yes	2005, Biham et al. [8] 2005, Wang et al.[10]	2^{51} 2^{39}
SHA-1	160	With flaws	2005, Wang et al. [11], 2006, Cannière et al.[1] [21],	2^{69} 2^{44}

[1] : The given complexity is for reduced(70) round of SHA-1.

TABLE I: Comparison of the Tttacks to the MD-SHA Family Hash Functions

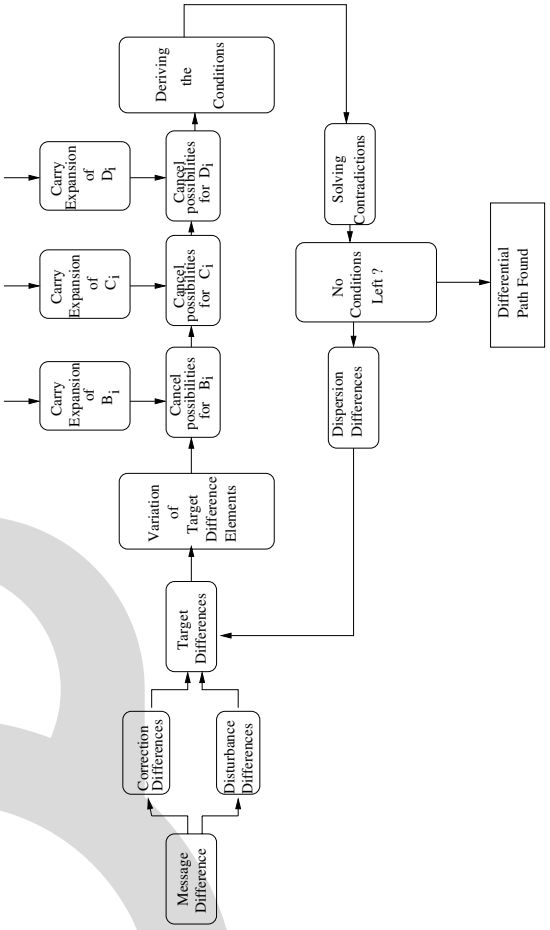


Fig. 3: The Algorithm of Schläffer et al.

REFERENCES

[1] Bart Preneel. Hash functions: Present state of the art. June 2005.
[2] Bart PRENEEL. Analysis and design of cryptographic hash functions. February 1993.
[3] X. Lai and J.L. Massey. Hash functions based on block ciphers. In *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658, R.A. Rueppel, Ed.*, pages 55-70, Springer-Verlag 1993.
[4] R. Merkle. One way hash functions and des. pages 428-446, Springer Verlag, 1990.
[5] I.B. Damgaard. A design principle for hash functions. pages 416-427, Springer Verlag, 1990.
[6] Florent Chabaud and Antoine Joux. Differential collisions in sha-0.
[7] Eli Biham and Rafi Chen. Near-collisions of sha-0. In *CRYPTO*, pages 290–305, 2004.
[8] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of sha-0 and reduced sha-1. In *EUROCRYPT*, pages 36–57, 2005.
[9] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions md4 and ripemd. In *EUROCRYPT*, pages 1–18, 2005.
[10] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on sha-0. In *CRYPTO*, pages 1–16, 2005.
[11] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In *CRYPTO*, pages 17–36, 2005.
[12] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *EUROCRYPT*, pages 19–35, 2005.
[13] Ron Rivest. The md4 message-digest algorithm. 1990.
[14] Ron Rivest. The md5 message-digest algorithm. 1992.
[15] National Institute of Standards and Technologies. Secure hash standard. In *Federal Information Processing Standards Publication, FIPS-180*, May 1993.
[16] National Institute of Standards and Technologies. Secure hash standard. In *Federal Information Processing Standards Publication, FIPS-180-1*, April 1995.
[17] Martin Schläffer. Cryptanalysis of md4. Master's thesis, Graz University of Technology, Graz,Austria, 2006.
[18] Yu Sasaki, Lei Wang, Kazuo Ohta, and Noboru Kunihiro. New message difference for md4. In *FSE*, pages 329–348, 2007.
[19] John Black, Martin Cochran, and Trevor Highland. A study of the md5 attacks: Insights and improvements. In *FSE*, pages 262–277, 2006.
[20] Martin Schläffer and Elisabeth Oswald. Searching for differential paths in md4. In *FSE*, pages 242–261, 2006.
[21] Christophe De Cannière and Christian Rechberger. Finding sha-1 characteristics: General results and applications. In *ASIACRYPT*, pages 1–20, 2006.
[22] Florent Chabaud and Antoine Joux. Differential collisions in sha-0. In *CRYPTO*, pages 56–71, 1998.
[23] Eli Biham and Rafi Chen. Near-collisions of sha-0. In *CRYPTO*, pages 290–305, 2004.